



```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             S
LL          II             S
LL          II             S
LL          II             S
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```

```
1 0001 0 MODULE FOR$$UDF_WN (%TITLE 'FOR$$UDF_WN - FORTRAN WRITE NAMELIST UDF Level'
2 0002 0 IDENT = '1-005' ! File: FORUDFWN.B32 Edit: SBL1005
3 0003 0 ) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1
30 0030 1 ++
31 0031 1 FACILITY: FORTRAN Compiled Code Support
32 0032 1
33 0033 1 ABSTRACT:
34 0034 1
35 0035 1 This module contains the User Data Formatter routines to perform
36 0036 1 FORTRAN NAMELIST WRITE statements.
37 0037 1
38 0038 1 ENVIRONMENT: Runs at any access mode - AST reentrant
39 0039 1
40 0040 1 AUTHOR: Steven B. Lionel, CREATION DATE: 29-August-1980
41 0041 1
42 0042 1 MODIFIED BY:
43 0043 1
44 0044 1 1-001 - Original. SBL 29-August-1980
45 0045 1 1-002 - Reflect group block spec change where count-of-variables is a word;
46 0046 1 second word is reserved. SBL 5-Dec-1980
47 0047 1 1-003 - Add text describing NAMELIST descriptor block. SBL 15-April-1981
48 0048 1 1-004 - REQUIRE FORERR.B32 instead of external reference to FOR$K_ symbols.
49 0049 1 SBL 12-Aug-1981
50 0050 1 1-005 - Add ability to dump just the variable names for "?" feature.
51 0051 1 Use prologue file. SBL 24-May-1983
52 0052 1 --
53 0053 1
```



```
55 0054 1 %SBTTL 'Declarations'
56 0055 1
57 0056 1 PROLOGUE FILE:
58 0057 1
59 0058 1
60 0059 1 REQUIRE 'RTLIN:FORPROLOG'; ! FOR$ definitions
61 0125 1
62 0126 1
63 0127 1 TABLE OF CONTENTS:
64 0128 1
65 0129 1
66 0130 1 FORWARD ROUTINE
67 0131 1 FOR$$UDF_WN0: JSB_UDF0 NOVALUE, ! Start WRITE NAMELIST
68 0132 1 FOR$$DO_RML_OUTPUT: CALL_CCB NOVALUE, ! Do bulk of processing
69 0133 1 FOR$$UDF_WN9: JSB_UDF9 NOVALUE, ! End WRITE NAMELIST
70 0134 1 CHECK_FIELD: CALL_CCB; ! Check field width
71 0135 1
72 0136 1
73 0137 1 MACROS:
74 0138 1
75 0139 1 NONE
76 0140 1
77 0141 1 EQUATED SYMBOLS:
78 0142 1
79 0143 1 NONE
80 0144 1
81 0145 1 FIELDS:
82 0146 1
83 0147 1 NONE
84 0148 1
85 0149 1 OWN STORAGE:
86 0150 1
87 0151 1 NONE
88 0152 1
89 0153 1 EXTERNAL REFERENCES:
90 0154 1
91 0155 1
92 0156 1 EXTERNAL ROUTINE
93 0157 1 FOR$$REC_WSNO: JSB_REC0 NOVALUE, ! Set up for a write
94 0158 1 FOR$$REC_WSN1: JSB_REC1 NOVALUE, ! Write a record
95 0159 1 FOR$$UDF_WL1: CALL_CCB NOVALUE, ! Convert and move to buffer
96 0160 1 FOR$$SIGNAL_STO: NOVALUE; ! Signal fatal error
```

```

98 0161 1 %SBTTL 'FOR$$UDF_WNO - Start WRITE NAMELIST'
99 0162 1 GLOBAL ROUTINE FOR$$UDF_WNO: JSB_UDFO NOVALUE          ! Start WRITE NAMELIST
100 0163 1 =
101 0164 1
102 0165 1 ++
103 0166 1 FUNCTIONAL DESCRIPTION:
104 0167 1
105 0168 1 This routine starts a FORTRAN WRITE NAMELIST. It calls FOR$$DO_NML_OUTPUT
106 0169 1 to do the bulk of the work. There is no UDF1 routine in this module
107 0170 1 because WRITE NAMELIST statements have no I/O lists.
108 0171 1
109 0172 1 CALLING SEQUENCE:
110 0173 1
111 0174 1 JSB FOR$$UDF_WNO
112 0175 1
113 0176 1 FORMAL PARAMETERS:
114 0177 1
115 0178 1 NONE
116 0179 1
117 0180 1 IMPLICIT INPUTS:
118 0181 1
119 0182 1 CCB ! Register pointer to RAB/LUB/ISB
120 0183 1
121 0184 1 IMPLICIT OUTPUTS:
122 0185 1
123 0186 1 See FOR$$DO_NML_OUTPUT
124 0187 1
125 0188 1 COMPLETION STATUS:
126 0189 1
127 0190 1 NONE
128 0191 1
129 0192 1 SIDE EFFECTS:
130 0193 1
131 0194 1 See FOR$$DO_NML_OUTPUT
132 0195 1
133 0196 1 --
134 0197 1
135 0198 2 BEGIN
136 0199 2
137 0200 2 EXTERNAL REGISTER
138 0201 2 CCB = 11: REF $FOR$CCB_DECL;
139 0202 2
140 0203 2 FOR$$REC_WSNO ();
141 0204 2
142 0205 2 FOR$$DO_NML_OUTPUT (0); ! Indicate that both names and values should print
143 0206 2
144 0207 2 RETURN;
145 0208 2
146 0209 1 END; ! End of routine

```

```

.TITLE FOR$$UDF_WN FOR$$UDF_WN - FORTRAN WRITE NAMELIST
       T UDF Level
.IDENT \1-005\
.EXTRN FOR$$REC_WSNO, FOR$$REC_WSNO1
.EXTRN FOR$$UDF_WL1, FOR$$SIGNAL_STO

```

FOR\$\$UDF\_WN  
1-005

FOR\$\$UDF\_WN - FORTRAN WRITE NAMELIST UDF Level  
FOR\$\$UDF\_WNO - Start WRITE NAMELIST

1 16  
16-Sep-1984 00:53:55  
14-Sep-1984 12:32:56

VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORUDFWN.B32;1

Page 4  
(3)

.PSECT \_FOR\$CODE,NOWRT, SHR, PIC,2

00000000G 00 16 0000 FOR\$\$UDF\_WNO::

0000V CF 7E D4 00006  
01 FB 00008  
05 0000D

JSB FOR\$\$REC\_WSNO  
CLRL -(SP)  
CALLS #1, FOR\$\$DO\_NML\_OUTPUT  
RSB

: 0203  
: 0205  
:  
: 0209

; Routine Size: 14 bytes, Routine Base: \_FOR\$CODE + 0000



```

: 148 0210 1 %SBTTL 'FOR$$DO_NML_OUTPUT - Do WRITE NAMELIST'
: 149 0211 1 GLOBAL ROUTINE FOR$$DO_NML_OUTPUT (
: 150 0212 1 NAMES_ONLY ! Set if only names are to be printed
: 151 0213 1 ): CALL_CCB NOVALUE =
: 152 0214 1
: 153 0215 1 ++
: 154 0216 1 FUNCTIONAL DESCRIPTION:
: 155 0217 1
: 156 0218 1 This routine performs one WRITE NAMELIST statement.
: 157 0219 1
: 158 0220 1 CALLING SEQUENCE:
: 159 0221 1
: 160 0222 1 CALL FOR$$DO_NML_OUTPUT (NAMES_ONLY.rl.v)
: 161 0223 1
: 162 0224 1 FORMAL PARAMETERS:
: 163 0225 1
: 164 0226 1 NAMES_ONLY 0 if both names and values are to be printed
: 165 0227 1 1 if only names are to be printed. This can
: 166 0228 1 occur if called from FOR$$UDF_RN to satisfy
: 167 0229 1 a "?" inquiry.
: 168 0230 1
: 169 0231 1 IMPLICIT INPUTS:
: 170 0232 1
: 171 0233 1 CCB ! Register pointer to RAB/LUB/ISB
: 172 0234 1 ISB$A_FMT_BEG ! Address of NAMELIST group descriptor
: 173 0235 1
: 174 0236 1 IMPLICIT OUTPUTS:
: 175 0237 1
: 176 0238 1
: 177 0239 1 COMPLETION STATUS:
: 178 0240 1
: 179 0241 1 NONE
: 180 0242 1
: 181 0243 1 SIDE EFFECTS:
: 182 0244 1
: 183 0245 1
: 184 0246 1 --
: 185 0247 1
: 186 0248 1 !<BLF/PAGE>
```

```
188 0249 1  !+
189 0250 1  Each NAMELIST descriptor block has the following form:
190 0251 1
191 0252 1      3 3 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1
192 0253 1      1 0 9 8 7 6 5 4 3 2 1 0 9 8 7 6 5 4 3 2 1 0
193 0254 1
194 0255 1
195 0256 1  0  +-----+
196 0257 1      | Address of ASCII name of NAMELIST group |
197 0258 1  1  +-----+
198 0259 1      | Reserved | Count of NAMELIST variables |
199 0260 1  2  +-----+
200 0261 1      | Address of ASCII name of variable 1 |
201 0262 1  3  +-----+
202 0263 1      | Address of standard VAX descriptor for variable 1 |
203 0264 1  4  +-----+
204 0265 1      | ... |
205 0266 1  5  +-----+
206 0267 1      | Address of ASCII name of variable n |
207 0268 1  6  +-----+
208 0269 1      | Address of standard VAX descriptor for variable n |
209 0270 1
210 0271 1
211 0272 1  The NAMELIST group name and the variable names which are pointed to in
212 0273 1  the NAMELIST descriptor block are upper case only. The FORTRAN
213 0274 1  compiler or other calling program is responsible for case conversion
214 0275 1  of the name strings. In NAMELIST input data, case is significant only
215 0276 1  in character literals. The run-time library is responsible for case
216 0277 1  conversion of NAMELIST input data.
217 0278 1
218 0279 1  The allowable data types in variable descriptors are BU (BYTE), WU,
219 0280 1  LU, W, L, F, D, G, H, T, FC, DC, and GC. The allowable descriptor
220 0281 1  classes are scalar and array. For the array class descriptor, the
221 0282 1  descriptor flags COLUMN, COEFF, and BOUNDS must be set, indicating
222 0283 1  column-major order and the presence of coefficient and bounds blocks.
223 0284 1  The number of dimensions must not exceed 7.
224 0285 1  --
225 0286 1
226 0287 1  !<BLF/PAGE>
227 0288 2  BEGIN
228 0289 2
229 0290 2  EXTERNAL REGISTER
230 0291 2  CCB = 11: REF $FOR$CCB_DECL;
231 0292 2
232 0293 2  LOCAL
233 0294 2  GROUP: REF VECTOR [, LONG],      ! NAMELIST group descriptor
234 0295 2  NVAR$,                          ! Number of variables in group
235 0296 2  VALUE_ADDR: REF VECTOR [, BYTE]; ! Address of value
236 0297 2
237 0298 2  !+
238 0299 2  ! Write out group name
239 0300 2  !-
240 0301 2
241 0302 2  GROUP = .CCB [ISB$A_FMT_BEG];
242 0303 2  VALUE_ADDR = .GROUP[0];      ! Address of group name counted string
243 0304 2  IF NOT CHECK_FIELD (.VALUE_ADDR [0] + 2) ! Include leading "$"
244 0305 2  THEN
```



```
245 0306 3 BEGIN
246 0307 FOR$$SIGNAL_STO (FOR$K_OUTSTAOVE);
247 0308 RETURN;
248 0309 END;
249 0310 CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]); ! Write leading space
250 0311 CH$WCHAR_A (%C'$', CCB [LUB$A_BUF_PTR]); ! Write leading $
251 0312 CCB [LUB$A_BUF_PTR] = CH$MOVE (.VALUE_ADDR [0], VALUE_ADDR [1], .CCB [LUB$A_BUF_PTR]);
252 0313 FOR$$REC_WSN1 ?);
253 0314
254 0315 !+
255 0316 ! Scan through NAMELIST group and write all variables to the output stream.
256 0317 !-
257 0318
258 0319
259 0320 DECR NVAR$ FROM (. (GROUP [1]) < 0, 16, 0> - 1) TO 0 DO
260 0321 BEGIN
261 0322
262 0323 LOCAL
263 0324 OUT_NAME_LEN; ! Output name length
264 0325
265 0326 GROUP = GROUP [2]; ! Skip to next variable
266 0327 VALUE_ADDR = .GROUP [0]; ! Address of variable name counted string
267 0328
268 0329 !+
269 0330 ! Compute output name length so that the names are padded to lengths
270 0331 ! of 7, 15, 23, etc. Add a space for before the '='.
271 0332 !-
272 0333
273 0334 OUT_NAME_LEN = .VALUE_ADDR [0];
274 0335 IF NOT .NAMES_ONLY
275 0336 THEN
276 0337 OUT_NAME_LEN = .OUT_NAME_LEN + (8 - (.VALUE_ADDR [0] MOD 8));
277 0338
278 0339 IF NOT CHECK_FIELD (.OUT_NAME_LEN + 2) ! Include leading space, trailing '='
279 0340 THEN
280 0341 BEGIN
281 0342 FOR$$SIGNAL_STO (FOR$K_OUTSTAOVE);
282 0343 RETURN;
283 0344 END;
284 0345
285 0346 !+
286 0347 ! Write out variable name
287 0348 !-
288 0349
289 0350 CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
290 0351 CCB [LUB$A_BUF_PTR] = CH$COPY (.VALUE_ADDR [0], VALUE_ADDR [1],
291 0352 %C' ', .OUT_NAME_LEN, .CCB [LUB$A_BUF_PTR]);
292 0353
293 0354 !+
294 0355 ! Only print values if NAMES_ONLY is false.
295 0356 !-
296 0357
297 0358 IF NOT .NAMES_ONLY
298 0359 THEN
299 0360 BEGIN
300 0361 CH$WCHAR_A (%C'=', CCB [LUB$A_BUF_PTR]);
301 0362
```

```
302 0363 4      !+
303 0364 4      !- Output all values in variable
304 0365 4
305 0366 4
306 0367 5      BEGIN
307 0368 5      LOCAL
308 0369 5      VAR_DESC: REF BLOCK [, BYTE],      ! Variable descriptor
309 0370 5      CUR_ADR,      ! Current variable address
310 0371 5      END_ADR,      ! End of variable
311 0372 5      ELEM_TYPE,      ! Element datatype passed to FOR$$UDF_WL1
312 0373 5      CMPLX_FLAG;      ! Complex flag passed to FOR$$UDF_WL1
313 0374 5      VAR_DESC = .GROUP [1];      ! Get descriptor
314 0375 5      CUR_ADR = .VAR_DESC [DSC$A_POINTER];
315 0376 5      IF .VAR_DESC [DSC$B_CLASS] EQL DSC$K_CLASS_A
316 0377 5      THEN
317 0378 5          END_ADR = .CUR_ADR + .VAR_DESC [DSC$L_ARSIZE]
318 0379 5      ELSE
319 0380 5          END_ADR = .CUR_ADR + .VAR_DESC [DSC$W_LENGTH];
320 0381 5      SELECT ONE .VAR_DESC [DSC$B_DTYPE] OF
321 0382 5          SET
322 0383 5          [DSC$K_DTYPE_FC]:
323 0384 6              BEGIN
324 0385 6                  ELEM_TYPE = DSC$K_DTYPE_F;
325 0386 6                  CMPLX_FLAG = 0;
326 0387 5                  END;
327 0388 5          [DSC$K_DTYPE_DC]:
328 0389 6              BEGIN
329 0390 6                  ELEM_TYPE = DSC$K_DTYPE_D;
330 0391 6                  CMPLX_FLAG = 0;
331 0392 5                  END;
332 0393 5          [DSC$K_DTYPE_GC]:
333 0394 6              BEGIN
334 0395 6                  ELEM_TYPE = DSC$K_DTYPE_G;
335 0396 6                  CMPLX_FLAG = 0;
336 0397 5                  END;
337 0398 5          [OTHERWISE]:
338 0399 6              BEGIN
339 0400 6                  ELEM_TYPE = .VAR_DESC [DSC$B_DTYPE];
340 0401 6                  !+
341 0402 6                  ! FORTRAN passes us BU for B, so change it here.
342 0403 6                  IF .ELEM_TYPE EQL DSC$K_DTYPE_BU
343 0404 6                  THEN
344 0405 6                      ELEM_TYPE = DSC$K_DTYPE_B;
345 0406 6                      CMPLX_FLAG = 2;
346 0407 5                  END;
347 0408 5          TES;
348 0409 5      WHILE .END_ADR GTRA .CUR_ADR DO
349 0410 5          BEGIN
350 0411 6              LOCAL
351 0412 6              CUR_POS,
352 0413 6              REPEAT_COUNT;
353 0414 6              !+
354 0415 6              !- Build repeat count
355 0416 6
356 0417 6              REPEAT_COUNT = 1;
```

```

CUR_POS = .CUR_ADR + .VAR_DESC [DSC$W_LENGTH];
WHILE .CUR_POS - LSSA .END_ADR DO
  BEGIN
    IF NOT CH$EQL (.VAR_DESC [DSC$W_LENGTH],
                  .CUR_ADR,
                  .VAR_DESC [DSC$W_LENGTH],
                  .CUR_POS,
                  0)
    THEN
      EXITLOOP;
    CUR_POS = .CUR_POS + .VAR_DESC [DSC$W_LENGTH];
    REPEAT_COUNT = .REPEAT_COUNT + 1;
  END;

!+
! Is this variable of type CHARACTER? If so, do all the
! processing here. Otherwise, let FOR$UDF_WL1 do most of
! the work.
!-

IF .ELEM_TYPE EQL DSC$K_DTYPE_T
THEN
  BEGIN
    !+
    ! It's CHARACTER.
    !-

    LOCAL
      REPEAT_DSC: DSC$DESCRIPTOR, ! Repeat string descriptor
      REPEAT_STR: VECTOR [12, BYTE], ! Repeat string
      FAO_DSC: DSC$DESCRIPTOR; ! FAO control string descr

    !+
    ! Build repeat count string.
    !-

    REPEAT_DSC [DSC$W_LENGTH] = 0;
    IF .REPEAT_COUNT GTR 1
    THEN
      BEGIN
        REPEAT_DSC [DSC$W_LENGTH] = 12;
        REPEAT_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
        REPEAT_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
        REPEAT_DSC [DSC$A_POINTER] = REPEAT_STR;
        FAO_DSC [DSC$A_POINTER] = UPLIT BYTE ('!SL*');
        FAO_DSC [DSC$W_LENGTH] = %CHARCOUNT ('!SL*');
        FAO_DSC [DSC$B_DTYPE] = DSC$K_DTYPE_T;
        FAO_DSC [DSC$B_CLASS] = DSC$K_CLASS_S;
        $FAO (FAO_DSC, ! Control string
              REPEAT_DSC [DSC$W_LENGTH], ! Returned length
              REPEAT_DSC, ! Output string
              .REPEAT_COUNT);
      END;

    !+
    ! See if there is enough room for the repeat count
    !-

```

P  
P



FOR\$UDF\_WN  
1-005

FOR\$UDF\_WN - FORTRAN WRITE NAMELIST UDF Level  
FOR\$DO\_RML\_OUTPUT - Do WRITE NAMELIST

C 1  
16-Sep-1984 00:53:55  
14-Sep-1984 12:32:56

VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORUDFWN.B32;1

Page 10  
(5)

```
416 0477 7 IF NOT CHECK_FIELD (2 + .REPEAT_DSC [DSC$W_LENGTH])
417 0478 7 THEN
418 0479 8 BEGIN
419 0480 8 FOR$REC_WSN1 ();
420 0481 8 IF NOT CHECK_FIELD (2 + .REPEAT_DSC [DSC$W_LENGTH])
421 0482 8 THEN
422 0483 8 FOR$SIGNAL_STO (FOR$K_OUTSTADVE);
423 0484 7 END;
424 0485 7
425 0486 7
426 0487 7 !+ Write out a leading space, the repeat count and an
427 0488 7 !- initial apostrophe.
428 0489 7
429 0490 7
430 0491 7 CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
431 0492 7 CCB [LUB$A_BUF_PTR] = CH$MOVE (.REPEAT_DSC [DSC$W_LENGTH], REPEAT_STR,
432 0493 7 .CCB [LUB$A_BUF_PTR]);
433 0494 7 CH$WCHAR_A (%C''', CCB [LUB$A_BUF_PTR]);
434 0495 7
435 0496 7 !+
436 0497 7 !- Write out each character of the string, substituting two
437 0498 7 !- apostrophes for each apostrophe found in the string.
438 0499 7
439 0500 7
440 0501 7 INCR I FROM 1 TO .VAR_DESC [DSC$W_LENGTH] DO
441 0502 8 BEGIN
442 0503 8 IF NOT CHECK_FIELD (1)
443 0504 8 THEN
444 0505 9 BEGIN
445 0506 9 FOR$REC_WSN1 ();
446 0507 9 CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
447 0508 8 END;
448 0509 8 IF CH$RCHAR (.CUR_ADR) EQL %C'''
449 0510 8 THEN
450 0511 9 BEGIN
451 0512 9 CH$WCHAR_A (%C''', CCB [LUB$A_BUF_PTR]);
452 0513 9 IF NOT CHECK_FIELD (1)
453 0514 9 THEN
454 0515 10 BEGIN
455 0516 10 FOR$REC_WSN1 ();
456 0517 10 CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
457 0518 9 END;
458 0519 8 END;
459 0520 8 COPY_BYTE_A (CUR_ADR, CCB [LUB$A_BUF_PTR]);
460 0521 7 END;
461 0522 7
462 0523 7 !+
463 0524 7 !- Write out the closing apostrophe.
464 0525 7
465 0526 7
466 0527 7 IF NOT CHECK_FIELD (1)
467 0528 7 THEN
468 0529 8 BEGIN
469 0530 8 FOR$REC_WSN1 ();
470 0531 8 CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);
471 0532 7 END;
472 0533 7 CH$WCHAR_A (%C''', CCB [LUB$A_BUF_PTR]);
```

```
473      0534 7          CUR_ADR = .CUR_POS;  
474      0535 7          END  
475      0536 7  
476      0537 6      ELSE  
477      0538 6  
478      0539 7          BEGIN  
479      0540 7          |  
480      0541 7          | Not CHARACTER.  
481      0542 7          | Call list directed routine to output value.  
482      0543 7          |  
483      0544 7          |  
484      0545 7          | FOR$UDF_WL1 (.ELEM_TYPE, .VAR_DESC [DSC$W_LENGTH], .CUR_ADR,  
485      0546 7          | .CMPLX_FLAG, .REPEAT_COUNT);  
486      0547 7          CUR_ADR = .CUR_POS;  
487      0548 6          END;  
488      0549 6  
489      0550 6      |  
490      0551 6      | Put out a separating comma if values to come  
491      0552 6      |  
492      0553 6  
493      0554 6      IF .CUR_ADR LSSA .END_ADR  
494      0555 6      THEN  
495      0556 7          BEGIN  
496      0557 7          IF NOT CHECK_FIELD (1)  
497      0558 7          THEN  
498      0559 8              BEGIN  
499      0560 8              FOR$REC_WSN1 ();  
500      0561 8              CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);  
501      0562 7              END;  
502      0563 7              CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);  
503      0564 6              END;  
504      0565 5          END;  
505      0566 4      END;  
506      0567 4  
507      0568 4      |  
508      0569 4      | If this is not the last variable, write out a comma  
509      0570 4      |  
510      0571 4  
511      0572 4      IF .NVAR$ NEQ 0  
512      0573 4      THEN  
513      0574 5          BEGIN  
514      0575 5          IF NOT CHECK_FIELD (1)  
515      0576 5          THEN  
516      0577 6              BEGIN  
517      0578 6              FOR$REC_WSN1 ();  
518      0579 6              CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);  
519      0580 5              END;  
520      0581 5              CH$WCHAR_A (%C' ', CCB [LUB$A_BUF_PTR]);  
521      0582 4              END;  
522      0583 4  
523      0584 3          END;  
524      0585 3  
525      0586 3      |  
526      0587 3      | Write this record.  
527      0588 3      |  
528      0589 3  
529      0590 3      FOR$REC_WSN1 ();
```

```

530      0591      3
531      0592      3
532      0593      3
533      0594      3
534      0595      3
535      0596      3
536      0597      3
537      0598      3
538      0599      3
539      0600      3
540      0601      3
541      0602      3
542      0603      3
543      0604      3
544      0605      3
545      0606      3
546      0607      3
547      0608      3
548      0609      3

```

END;  
 !+  
 !- All variables written. Put out \$END block delimiter.  
 !-  
 IF NOT CHECK\_FIELD (5)  
 THEN  
 BEGIN  
 FOR\$\$SIGNAL\_STO (FOR\$K\_OUTSTAOVE);  
 RETURN;  
 END;  
 CCB [LUB\$A\_BUF\_PTR] = CH\$MOVE (5, UPLIT BYTE (' \$END'), .CCB [LUB\$A\_BUF\_PTR]);  
 FOR\$\$REC\_WSN1 7);  
 RETURN;  
 END;

! End of routine

! End of routine

PC	Op	OpC	OpD	OpE	OpF	OpG	OpH	OpI	OpJ	OpK	OpL	OpM	OpN	OpO	OpP	OpQ	OpR	OpS	OpT	OpU	OpV	OpW	OpX	OpY	OpZ	OpAA	OpAB	OpAC	OpAD	OpAE	OpAF	OpAG	OpAH	OpAI	OpAJ	OpAK	OpAL	OpAM	OpAN	OpAO	OpAP	OpAQ	OpAR	OpAS	OpAT	OpAU	OpAV	OpAW	OpAX	OpAY	OpAZ	OpBA	OpBB	OpBC	OpBD	OpBE	OpBF	OpBG	OpBH	OpBI	OpBJ	OpBK	OpBL	OpBM	OpBN	OpBO	OpBP	OpBQ	OpBR	OpBS	OpBT	OpBU	OpBV	OpBW	OpBX	OpBY	OpBZ	OpCA	OpCB	OpCC	OpCD	OpCE	OpCF	OpCG	OpCH	OpCI	OpCJ	OpCK	OpCL	OpCM	OpCN	OpCO	OpCP	OpCQ	OpCR	OpCS	OpCT	OpCU	OpCV	OpCW	OpCX	OpCY	OpCZ	OpDA	OpDB	OpDC	OpDD	OpDE	OpDF	OpDG	OpDH	OpDI	OpDJ	OpDK	OpDL	OpDM	OpDN	OpDO	OpDP	OpDQ	OpDR	OpDS	OpDT	OpDU	OpDV	OpDW	OpDX	OpDY	OpDZ	OpEA	OpEB	OpEC	OpED	OpEE	OpEF	OpEG	OpEH	OpEI	OpEJ	OpEK	OpEL	OpEM	OpEN	OpEO	OpEP	OpEQ	OpER	OpES	OpET	OpEU	OpEV	OpEW	OpEX	OpEY	OpEZ	OpFA	OpFB	OpFC	OpFD	OpFE	OpFF	OpFG	OpFH	OpFI	OpFJ	OpFK	OpFL	OpFM	OpFN	OpFO	OpFP	OpFQ	OpFR	OpFS	OpFT	OpFU	OpFV	OpFW	OpFX	OpFY	OpFZ	OpGA	OpGB	OpGC	OpGD	OpGE	OpGF	OpGG	OpGH	OpGI	OpGJ	OpGK	OpGL	OpGM	OpGN	OpGO	OpGP	OpGQ	OpGR	OpGS	OpGT	OpGU	OpGV	OpGW	OpGX	OpGY	OpGZ	OpHA	OpHB	OpHC	OpHD	OpHE	OpHF	OpHG	OpHH	OpHI	OpHJ	OpHK	OpHL	OpHM	OpHN	OpHO	OpHP	OpHQ	OpHR	OpHS	OpHT	OpHU	OpHV	OpHW	OpHX	OpHY	OpHZ	OpIA	OpIB	OpIC	OpID	OpIE	OpIF	OpIG	OpIH	OpII	OpIJ	OpIK	OpIL	OpIM	OpIN	OpIO	OpIP	OpIQ	OpIR	OpIS	OpIT	OpIU	OpIV	OpIW	OpIX	OpIY	OpIZ	OpJA	OpJB	OpJC	OpJD	OpJE	OpJF	OpJG	OpJH	OpJI	OpJJ	OpJK	OpJL	OpJM	OpJN	OpJO	OpJP	OpJQ	OpJR	OpJS	OpJT	OpJU	OpJV	OpJW	OpJX	OpJY	OpJZ	OpKA	OpKB	OpKC	OpKD	OpKE	OpKF	OpKG	OpKH	OpKI	OpKJ	OpKK	OpKL	OpKM	OpKN	OpKO	OpKP	OpKQ	OpKR	OpKS	OpKT	OpKU	OpKV	OpKW	OpKX	OpKY	OpKZ	OpLA	OpLB	OpLC	OpLD	OpLE	OpLF	OpLG	OpLH	OpLI	OpLJ	OpLK	OpLL	OpLM	OpLN	OpLO	OpLP	OpLQ	OpLR	OpLS	OpLT	OpLU	OpLV	OpLW	OpLX	OpLY	OpLZ	OpMA	OpMB	OpMC	OpMD	OpME	OpMF	OpMG	OpMH	OpMI	OpMJ	OpMK	OpML	OpMN	OpMO	OpMP	OpMQ	OpMR	OpMS	OpMT	OpMU	OpMV	OpMW	OpMX	OpMY	OpMZ	OpNA	OpNB	OpNC	OpND	OpNE	OpNF	OpNG	OpNH	OpNI	OpNJ	OpNK	OpNL	OpNM	OpNN	OpNO	OpNP	OpNQ	OpNR	OpNS	OpNT	OpNU	OpNV	OpNW	OpNX	OpNY	OpNZ	OpOA	OpOB	OpOC	OpOD	OpOE	OpOF	OpOG	OpOH	OpOI	OpOJ	OpOK	OpOL	OpOM	OpON	OpOO	OpOP	OpOQ	OpOR	OpOS	OpOT	OpOU	OpOV	OpOW	OpOX	OpOY	OpOZ	OpPA	OpPB	OpPC	OpPD	OpPE	OpPF	OpPG	OpPH	OpPI	OpPJ	OpPK	OpPL	OpPM	OpPN	OpPO	OpPP	OpPQ	OpPR	OpPS	OpPT	OpPU	OpPV	OpPW	OpPX	OpPY	OpPZ	OpQA	OpQB	OpQC	OpQD	OpQE	OpQF	OpQG	OpQH	OpQI	OpQJ	OpQK	OpQL	OpQM	OpQN	OpQO	OpQP	OpQQ	OpQR	OpQS	OpQT	OpQU	OpQV	OpQW	OpQX	OpQY	OpQZ	OpRA	OpRB	OpRC	OpRD	OpRE	OpRF	OpRG	OpRH	OpRI	OpRJ	OpRK	OpRL	OpRM	OpRN	OpRO	OpRP	OpRQ	OpRR	OpRS	OpRT	OpRU	OpRV	OpRW	OpRX	OpRY	OpRZ	OpSA	OpSB	OpSC	OpSD	OpSE	OpSF	OpSG	OpSH	OpSI	OpSJ	OpSK
----	----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	-----	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------	------



52			08	A0	9E	0006C	MOVAB	8(R0), OUT_NAME_LEN	...	
			02	A2	9F	00070	PUSHAB	2(OUT_NAME_LEN)	...	0339
	0000V	CF		01	FB	00073	CALLS	#1, CHECK_FIELD	...	
		03		50	E8	00078	BLBS	R0, 4\$	...	
				02	13	31	BRW	35\$	...	
		56	B0	AB	9E	0007E	MOVAB	-80(CCB), R6	...	0350
	00	B6		20	90	00082	MOVB	#32, @0(R6)	...	
				66	D6	00086	INCL	(R6)	...	
		50		6A	9A	00088	MOVZBL	(VALUE_ADDR), R0	...	0351
52	20	01	AA	50	2C	0008B	MOVC5	R0, 1(VALUE_ADDR), #32, OUT_NAME_LEN, -	...	0352
				00	B6	00091		@0(R6)	...	
		66		53	D0	00093	MOVL	R3, (R6)	...	
		03	18	AE	E8	00096	BLBS	24(SP), 5\$	...	0358
				01	DB	31	BRW	31\$	...	
	00	B6		3D	90	0009D	MOVB	#61, @0(R6)	...	0361
				66	D6	000A1	INCL	(R6)	...	
	50	6E		04	C1	000A3	ADDL3	#4, GROUP, R0	...	0374
		57		60	D0	000A7	MOVL	(R0), VAR_DESC	...	
		59	04	A7	D0	000AA	MOVL	4(VAR_DESC), CUR_ADR	...	0375
		04	03	A7	91	000AE	CMPB	3(VAR_DESC), #4	...	0376
				08	12	000B2	BNEQ	6\$	...	
	10	AE	0C	B7	49	9E	MOVAB	@12(VAR_DESC)[CUR_ADR], END_ADR	...	0378
				08	11	000BA	BRB	7\$	...	
		50		67	3C	000BC	MOVZWL	(VAR_DESC), R0	...	0380
10	AE	59		50	C1	000BF	ADDL3	R0, CUR_ADR, END_ADR	...	
		50	02	A7	9A	000C4	MOVZBL	2(VAR_DESC), R0	...	0381
		0C		50	91	000C8	CMPB	R0, #T2	...	0383
				05	12	000CB	BNEQ	8\$	...	
		58		0A	D0	000CD	MOVL	#10, ELEM_TYPE	...	0385
				12	11	000D0	BRB	10\$	...	0386
		0D		50	91	000D2	CMPB	R0, #13	...	0388
				05	12	000D5	BNEQ	9\$	...	
		58		08	D0	000D7	MOVL	#11, ELEM_TYPE	...	0390
				08	11	000DA	BRB	10\$	...	0391
		1D		50	91	000DC	CMPB	R0, #29	...	0393
				08	12	000DF	BNEQ	11\$	...	
		58		1B	D0	000E1	MOVL	#27, ELEM_TYPE	...	0395
			0C	AE	D4	000E4	CLRL	CMPLX_FLAG	...	0396
				0F	11	000E7	BRB	13\$	...	0381
		58		50	D0	000E9	MOVL	R0, ELEM_TYPE	...	0400
		02		58	D1	000EC	CMPL	ELEM_TYPE, #2	...	0403
				03	12	000EF	BNEQ	12\$	...	
		58		06	D0	000F1	MOVL	#6, ELEM_TYPE	...	0405
	0C	AE		02	D0	000F4	MOVL	#2, CMPLX_FLAG	...	0406
		59	10	AE	D1	000F8	CMPL	END_ADR, CUR_ADR	...	0410
				03	1A	000FC	BGTRU	14\$	...	
				01	54	31	BRW	29\$	...	
		54		01	D0	00101	MOVL	#1, REPEAT_COUNT	...	0419
	04	AE		67	3C	00104	MOVZWL	(VAR_DESC), 4(SP)	...	0420
	08	AE	04	BE	49	9E	MOVAB	@4(SP)[CUR_ADR], CUR_POS	...	
	10	AE	08	AE	D1	0010E	CMPL	CUR_POS, END_ADR	...	0421
				11	1E	00113	BGEQU	16\$	...	
08	BE			AE	29	00115	CMPC3	4(SP), (CUR_ADR), @CUR_POS	...	0423
		69	04	09	12	0011B	BNEQ	16\$	...	
				AE	C0	0011D	ADDL2	4(SP), CUR_POS	...	0430
	08	AE	04	54	D6	00122	INCL	REPEAT_COUNT	...	0431
				E8	11	00124	BRB	15\$	...	0421

OE	58	D1	00126	16\$:	CMPL	ELEM_TYPE, #14	0440
	03	13	00129		BEQL	17\$	
	00E9	31	00128		BRW	25\$	
	30	AE	B4 0012E	17\$:	CLRW	REPEAT_DSC	0455
01	54	D1	00131		CMPL	REPEAT_COUNT, #1	0456
	2D	15	00134		BLEQ	18\$	
30	AE	010E000C	8F D0 00136		MOVL	#17694732, REPEAT_DSC	0459
34	AE	24	AE 9E 0013E		MOVAB	REPEAT_STR, REPEAT_DSC+4	0462
20	AE	FEB0	CF 9E 00143		MOVAB	P.AAA, -FAO_DSC+4	0463
1C	AE	010E0004	8F D0 00149		MOVL	#17694724, -FAO_DSC	0464
	54	DD	00151		PUSHL	REPEAT_COUNT	0470
	34	AE	9F 00153		PUSHAB	REPEAT_DSC	
	38	AE	9F 00156		PUSHAB	REPEAT_DSC	
	28	AE	9F 00159		PUSHAB	FAO_DSC	
00000000G	00	04	FB 0015C		CALLS	#4, -SYSSFAO	
	52	30	AE 3C 00163	18\$:	MOVZWL	REPEAT_DSC, R2	0477
	52	02	CO 00167		ADDL2	#2, R2	
		52	DD 0016A		PUSHL	R2	
0000V	CF	01	FB 0016C		CALLS	#1, CHECK_FIELD	
	1B	50	E8 00171		BLBS	R0, 19\$	
	00000000G	00	16 00174		JSB	FORSSREC_WSN1	0480
		52	DD 0017A		PUSHL	R2	0481
000CV	CF	01	FB 0017C		CALLS	#1, CHECK_FIELD	
	0B	50	E8 00181		BLBS	R0, 19\$	
	7E	42	8F 9A 00184		MOVZBL	#66, -(SP)	0483
00000000G	00	01	FB 00188		CALLS	#1, FORSSIGNAL_STO	
	56	80	AB 9E 0018F	19\$:	MOVAB	-80(CCB), R6	0491
00	B6	20	90 00193		MOVB	#32, a0(R6)	
		66	D6 00197		INCL	(R6)	
00	B6	24	AE 28 00199		MOV3	REPEAT_DSC, REPEAT_STR, a0(R6)	0493
		66	D0 001A0		MOVL	R3, (R6)	
	00	B6	27 90 001A3		MOVB	#39, a0(R6)	0494
		66	DE 001A7		INCL	(R6)	
		52	D4 001A9		CLRL	1	0501
		45	11 001AB		BRB	23\$	
		01	1D 001AD	20\$:	PUSHL	#1	0503
0000V	CF	01	FB 001AF		CALLS	#1, CHECK_FIELD	
	0D	50	E8 001B4		BLBS	R0, 21\$	
	00000000G	00	16 001B7		JSB	FORSSREC_WSN1	0506
B0	BB	20	90 001BD		MOVB	#32, a-80(CCB)	0507
		80	AB D6 001C1		INCL	-80(CCB)	
	27	69	91 001C4	21\$:	CMPB	(CUR_ADR), #39	0509
		1E	12 001C7		BNEQ	22\$	
B0	BB	27	90 001C9		MOVB	#39, a-80(CCB)	0512
		80	AB D6 001CD		INCL	-80(CCB)	
		01	DD 001D0		PUSHL	#1	0513
0000V	CF	01	FB 001D2		CALLS	#1, CHECK_FIELD	
	0D	50	E8 001D7		BLBS	R0, 22\$	
	00000000G	00	16 001DA		JSB	FORSSREC_WSN1	0516
B0	BB	20	90 001E0		MOVB	#32, a-80(CCB)	0517
		80	AB D6 001E4		INCL	-80(CCB)	
		80	AB D6 001E7	22\$:	INCL	-80(CCB)	0520
	50	80	AB D0 001EA		MOVL	-80(CCB), R0	
FF	A0	89	90 001EE		MOVB	(CUR_ADR)+-1(R0)	
B6	52	04	AE F3 001F2	23\$:	AOBLEQ	4(SPT, 1, 20\$	0501
		01	DD 001F7		PUSHL	#1	0527
0000V	CF	01	FB 001F9		CALLS	#1, CHECK_FIELD	

FOR\$\$UDF\_WN  
1-005

FOR\$\$UDF\_WN - FORTRAN WRITE NAMELIST UDF  
FOR\$\$DO\_NML\_OUTPUT - Do WRITE NAMELIST

Level 1  
16-Sep-1984 00:53:55  
14-Sep-1984 12:32:56

VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORUDFWN.B32;1

Page 15  
(5)

	OD		50	E8	001FE	BLBS	RO, 24\$	...	0530
		00000000G	00	16	00201	JSB	FOR\$\$REC_WSN1	...	0531
B0	BB		20	90	00207	MOVB	#32, a-80(CCB)	...	0533
		B0	AB	D6	0020B	INCL	-80(CCB)	...	0534
B0	BB		27	90	0020E	MOVB	#39, a-80(CCB)	...	0546
		B0	AB	D6	00212	INCL	-80(CCB)	...	0545
			13	11	00215	BRB	26\$	...	0547
			54	DD	00217	PUSHL	REPEAT_COUNT	...	0554
		10	AE	DD	00219	PUSHL	CMPLX_FLAG	...	0557
			59	DD	0021C	PUSHL	CUR_ADR	...	0560
		10	AE	DD	0021E	PUSHL	16(SP)	...	0561
			58	DD	00221	PUSHL	ELEM_TYPE	...	0563
00000000G	00		05	FB	00223	CALLS	#5, FOR\$\$UDF_WL1	...	0410
	59	08	AE	D0	0022A	MOVL	CUR_POS, CUR_ADR	...	0572
10	AE		59	D1	0022E	CMPL	CUR_ADR, END_ADR	...	0575
			1E	1E	00232	BGEQU	28\$	...	0578
			01	DD	00234	PUSHL	#1	...	0579
0000V	CF		01	FB	00236	CALLS	#1, CHECK_FIELD	...	0581
	OD		50	E8	0023B	BLBS	RO, 27\$	...	0590
		00000000G	00	16	0023E	JSB	FOR\$\$REC_WSN1	...	0320
B0	BB		20	90	00244	MOVB	#32, a-80(CCB)	...	0598
		B0	AB	D6	00248	INCL	-80(CCB)	...	0601
B0	BB		2C	90	0024B	MOVB	#44, a-80(CCB)	...	0600
		B0	AB	D6	0024F	INCL	-80(CCB)	...	0604
			FEA3	31	00252	BRW	13\$	...	0605
		14	AE	D5	00255	TSTL	NVARS	...	0609
			1E	13	00258	BEQL	31\$	...	
			01	DD	0025A	PUSHL	#1	...	
0000V	CF		01	FB	0025C	CALLS	#1, CHECK_FIELD	...	
	OD		50	E8	00261	BLBS	RO, 30\$	...	
		00000000G	00	16	00264	JSB	FOR\$\$REC_WSN1	...	
B0	BB		20	90	0026A	MOVB	#32, a-80(CCB)	...	
		B0	AB	D6	0026E	INCL	-80(CCB)	...	
B0	BB		2C	90	00271	MOVB	#44, a-80(CCB)	...	
		B0	AB	D6	00275	INCL	-80(CCB)	...	
		00000000G	00	16	00278	JSB	FOR\$\$REC_WSN1	...	
	02	14	AE	F4	0027E	SOBGEQ	NVARS, 33\$	...	
			03	11	00282	BRB	34\$	...	
			FDC6	31	00284	BRW	1\$	...	
			05	DD	00287	PUSHL	#5	...	
0000V	CF		01	FB	00289	CALLS	#1, CHECK_FIELD	...	
	OC		50	E8	0028E	BLBS	RO, 36\$	...	
	7E	42	8F	9A	00291	MOVZBL	#66, -(SP)	...	
00000000G	00		01	FB	00295	CALLS	#1, FOR\$\$SIGNAL_STO	...	
			04	0029C	RET			...	
B0	BB	FD59	05	28	0029D	MOVCS	#5, P.AAB, a-80(CCB)	...	
	B0	AB	53	D0	002A4	MOVL	R3, -80(CCB)	...	
		00000000G	00	16	002A8	JSB	FOR\$\$REC_WSN1	...	
			04	002AE	RET			...	

; Routine Size: 687 bytes, Routine Base: \_FOR\$CODE + 0017

; 549 0610 1 !<BLF/PAGE>



```
.. 551 0611 1 %SBTTL 'FOR$$UDF_WN9 - End WRITE NAMELIST'
.. 552 0612 1 GLOBAL ROUTINE FOR$$UDF_WN9: JSB_UDF9 NOVALUE      ! End WRITE NAMELIST
.. 553 0613 1 =
.. 554 0614 1
.. 555 0615 1 ++
.. 556 0616 1 FUNCTIONAL DESCRIPTION:
.. 557 0617 1
.. 558 0618 1     End a namelist-directed WRITE statement.  This procedure, although
.. 559 0619 1     a no-op, is necessary because FOR$IO_END dispatches to a UDF9 routine
.. 560 0620 1     based on the statement type.
.. 561 0621 1
.. 562 0622 1 CALLING SEQUENCE:
.. 563 0623 1
.. 564 0624 1     JSB FOR$$UDF_WN9
.. 565 0625 1
.. 566 0626 1 FORMAL PARAMETERS:
.. 567 0627 1
.. 568 0628 1     NONE
.. 569 0629 1
.. 570 0630 1 IMPLICIT INPUTS:
.. 571 0631 1
.. 572 0632 1     CCB                                ! Register pointer to RAB/LUB/ISB
.. 573 0633 1
.. 574 0634 1 IMPLICIT OUTPUTS:
.. 575 0635 1
.. 576 0636 1     NONE
.. 577 0637 1
.. 578 0638 1 COMPLETION STATUS: (or ROUTINE VALUE:)
.. 579 0639 1
.. 580 0640 1     NONE
.. 581 0641 1
.. 582 0642 1 SIDE EFFECTS:
.. 583 0643 1
.. 584 0644 1     NONE
.. 585 0645 1
.. 586 0646 1 --
.. 587 0647 1
.. 588 0648 2 BEGIN
.. 589 0649 2
.. 590 0650 2 RETURN;
.. 591 0651 2
.. 592 0652 1 END;                                ! End of routine FOR$$UDF_WN9
```

05 0000 FOR\$\$UDF\_WN9::  
RSB

: 0652

: Routine Size: 1 bytes.      Routine Base: \_FOR\$CODE + 02C6

: 593      0653 1 !<BLF/PAGE>

```
595 0654 1 %SBTTL 'CHECK_FIELD - Check field remaining for width'
596 0655 1 ROUTINE CHECK_FIELD (
597 0656 1     WIDTH
598 0657 1     ): CALL_CCB
599 0658 1     =
600 0659 1
601 0660 1 ++
602 0661 1 FUNCTIONAL DESCRIPTION:
603 0662 1
604 0663 1     Determine if there are sufficient characters remaining in the current
605 0664 1     record for a field of a specified width.
606 0665 1
607 0666 1 CALLING SEQUENCE:
608 0667 1
609 0668 1     status = CHECK_FIELD (width.rl.v)
610 0669 1
611 0670 1 FORMAL PARAMETERS:
612 0671 1
613 0672 1     width - The width of the field you wish to use
614 0673 1
615 0674 1 IMPLICIT INPUTS:
616 0675 1
617 0676 1     CCB                                ! Register pointer to RAB/LUB/ISB
618 0677 1
619 0678 1 IMPLICIT OUTPUTS:
620 0679 1
621 0680 1     NONE
622 0681 1
623 0682 1 COMPLETION STATUS: (or ROUTINE VALUE:)
624 0683 1
625 0684 1     1 if the field will fit, 0 otherwise
626 0685 1
627 0686 1 SIDE EFFECTS:
628 0687 1
629 0688 1     NONE
630 0689 1
631 0690 1 --
632 0691 1
633 0692 2 BEGIN
634 0693 2
635 0694 2 EXTERNAL REGISTER
636 0695 2     CCB = 11: REF $FOR$CCB_DECL;
637 0696 2
638 0697 2 RETURN ((.CCB [LUB$A_BUF_PTR] + .WIDTH) LEQA .CCB [LUB$A_BUF_END]);
639 0698 2
640 0699 1 END;                                ! End of routine CHECK_FIELD
```

```
0000 00000 CHECK_FIELD:
51      B0  AB      04  AC  C1 00002      .WORD  Save nothing
      B4  AB      50  D4 00008      ADDL3  WIDTH, -80(CCB), R1
      51  D1 0000A      CLRL   R0
      02  1A 0000E      CMPL   R1, -76(CCB)
      BGTRU  1$
```

```
0655
0697
...
```

FOR\$SUDF\_WN  
1-005

FOR\$SUDF\_WN - FORTRAN WRITE NAMELIST UDF Level 1  
CHECK\_FIELD - Check field remaining for width 14-Sep-1984 12:32:56

VAX-11 Bliss-32 V4.0-742  
[FORRTL.SRC]FORUDFWN.B32;1

Page 18  
(7)

50 D6 00010 INCL R0  
04 00012 1\$: RET

: 0699

; Routine Size: 19 bytes, Routine Base: \_FOR\$CODE + 02C7

; 641 0700 1 !<BLF/PAGE>



PSECT SUMMARY

Name	Bytes	Attributes
_FOR\$CODE	730	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
-\$255\$DUA28:[SYSLIB]STARLET.L32;1	9776	17	0	581	00:01.1
-\$255\$DUA28:[FORRTL.OBJ]FORLIB.L32;1	711	186	26	52	00:00.6
-\$255\$DUA28:[FORRTL.OBJ]RTLLIB.L32;1	36	0	0	8	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS\$:FORUDFWN/OBJ=OBJ\$:FORUDFWN MSRC\$:FORUDFWN/UPDATE=(ENHS:FORUDFWN)

Size: 721 code + 9 data bytes  
Run Time: 00:18.6  
Elapsed Time: 00:53.2  
Lines/CPU Min: 2264  
Lexemes/CPU-Min: 13809  
Memory Used: 274 pages  
Compilation Complete



0184 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY



0185 AH-BT13A-SE  
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION  
CONFIDENTIAL AND PROPRIETARY

